Learning the Java Language

(http://docs.oracle.com/javase/tutorial/java/index.html)

Objectives

- Study some fundamentals of Java languages: Data types, variables, arrays, operators, logic constructs.

- Pass arguments to the main method

- Input/output variables

- Concept: Package.

Keywords and Identifiers

- Keywords: Almost of them are similar to those in C language
- Naming Convention:

Letter $ _	Letters Digits, $ _

- Java is a case-sensitive language
- Identifiers must be different to keywords

Primitive Data Types - Variables

- A *primitive* is <u>a simple non-object</u> data type that represents a single value. Java's primitive data types are:

Type	Bytes	Minimum	Maximum
char	2	\u0000	\uFFFF
byte	1	-2^7	$2^7 - 1$
short	2	-2^{15}	$2^{15} - 1$
int	4	-2^{31}	$2^{31} - 1$
long	8	-2^{63}	$2^{63} - 1$
float	4		
double	8		
boolean	true/false		

Type   var [=Initial value] ;

Operators

Category (Descending Precedence)	Operators
Unary	`++ -- + - ! ~ (type)`
Arithmetic	`* / %` `+ -`
Shift	`<< >> >>>`
Comparison	`< <= > >= instanceof` `== !=`
Bitwise	`& ^ |`
Short-circuit	`&& ||`
Conditional	`?:`
Assignment	`= op=`

They are the same with those in C language

Using Operators Demonstration

```java
public class UseOps {
  public static void main(String[] args)
  {  int x=-1;
     System.out.println("-1<<1: " + (x<<1) );
     System.out.println("-1>>1: " + (x>>1) );
     System.out.println("-1>>>1: " + (x>>>1));
     System.out.println("3|4: " + (3|4));
     System.out.println("3&4: " + (3&4));
     System.out.println("3^4: " + (3^4));
     String S="Hello";
     boolean result= S instanceof String;
     System.out.println("Hello is an instance of String: " + result);
  }
}
```

```
Output - Chapter01 (run)
run:
-1<<1: -2
-1>>1: -1
-1>>>1: 2147483647
3|4: 7
3&4: 0
3^4: 7
Hello is an instance of String: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Using Operators Demonstration

Use 2 bytes to store value

```
Output - Chapter01 (run)
run:
-1<<1: -2
-1>>1: -1
-1>>>1: 2147483647
3|4: 7
3&4: 0
3^4: 7
Hello is an instance of String: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

1: ➜   0000 0000 0000 0001
             1111 1111 1111 1110 (1-complement)
-1 ➜   1111 1111 1111 1111 (2-complement)
-1 <<1 ➜   1111 1111 1111 1110 (-2)

-1 ➜   1111 1111 1111 1111
-1 >>1 ➜   1111 1111 1111 1111

-1 ➜   1111 1111 1111 1111
-1 >>>1 ➜ 0111 1111 1111 1111 (2147483647)

3 ➜   0000 0000 0000 0011
4 ➜   0000 0000 0000 0100
3|4 ➜   0000 0000 0000 0111 (7)

3 ➜   0000 0000 0000 0011
4 ➜   0000 0000 0000 0100
3&4 ➜   0000 0000 0000 0000 (0)

3 ➜   0000 0000 0000 0011
4 ➜   0000 0000 0000 0100
3^4 ➜   0000 0000 0000 0111 (7): XOR BIT

Literals and Value Variables

- Character: 'a'

- String: String S="Hello";

- Integral literals:

   28, 0x1c, 0X1A (default: int).
   123l, 123L (long)

- Floating point:

  1.234 (default: double)

  1.3f   1.3F

  1.3E+21

  1.3d   1.3D

Value variable

Stack

| n | 10 |
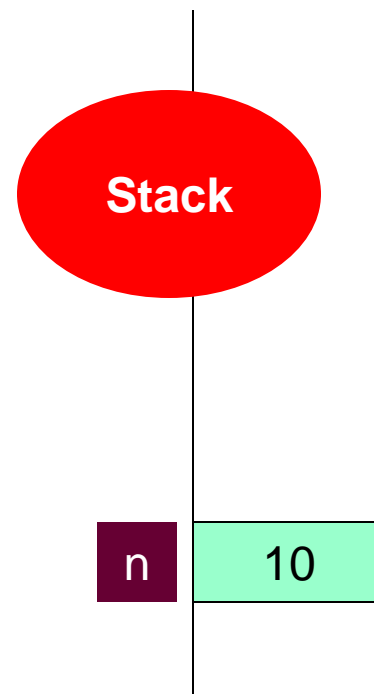
int n=10;

Literals and Value Variables

Escape Sequences

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a form feed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

Example Escape sequences

To print the sentence

She said "Hello!" to me.

you would write

System.out.println("She said \"Hello!\" to me.");

Java Expressions

- Java is an expression-oriented language. A simple expression in Java is either:
  - A constant: 7, false
  - A char - literal enclosed in single quotes: 'A', '3'
  - A String - literal enclosed in double quotes: "foo"
  - The name of any properly declared variables: x
  - Any two|one of the preceding types of expression that are combined with one of the Java binary operators: i++, x + 2, (x + 2)

One Dimensional Arrays (1)

- An *array* is a container object that holds a fixed number of values of a single type.

- The length of an array is established when the array is created.

- Each item in an array is called an *element*, and each element is accessed by its numerical *index.*
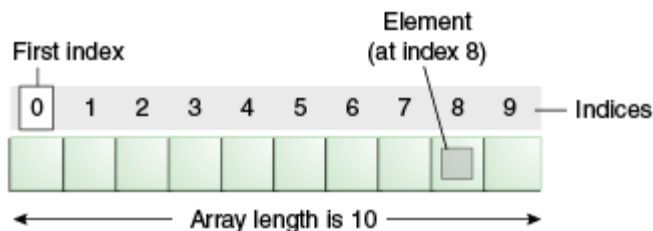
One Dimensional Arrays (2)

- Declaring a Variable to Refer to an Array

  ```
  int[] anArray;
  ```
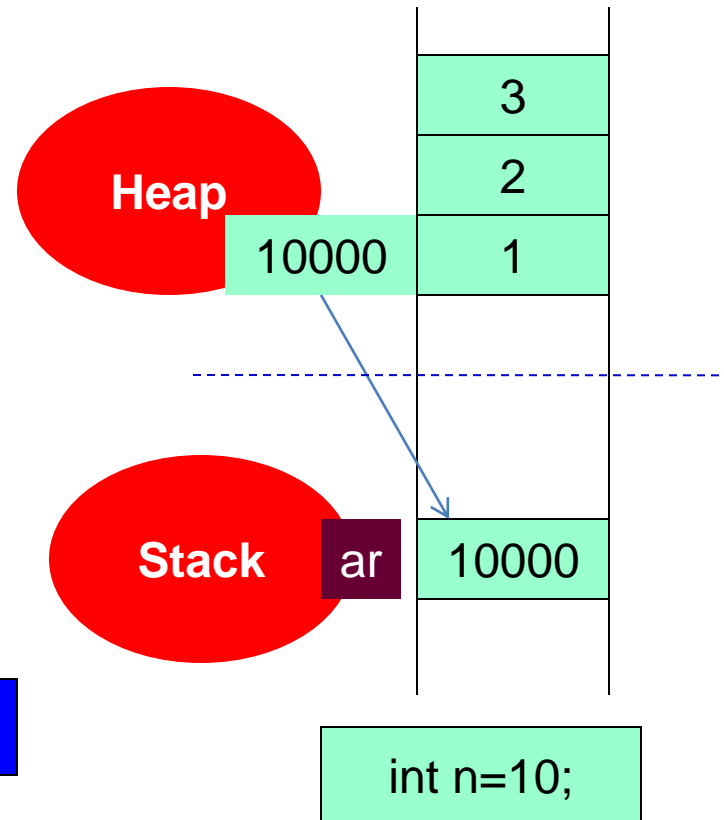 or `float anArrayOfFloats[];`

- Creating, Initializing, and Accessing an Array

  ```
  anArray = new int[10];
  ```

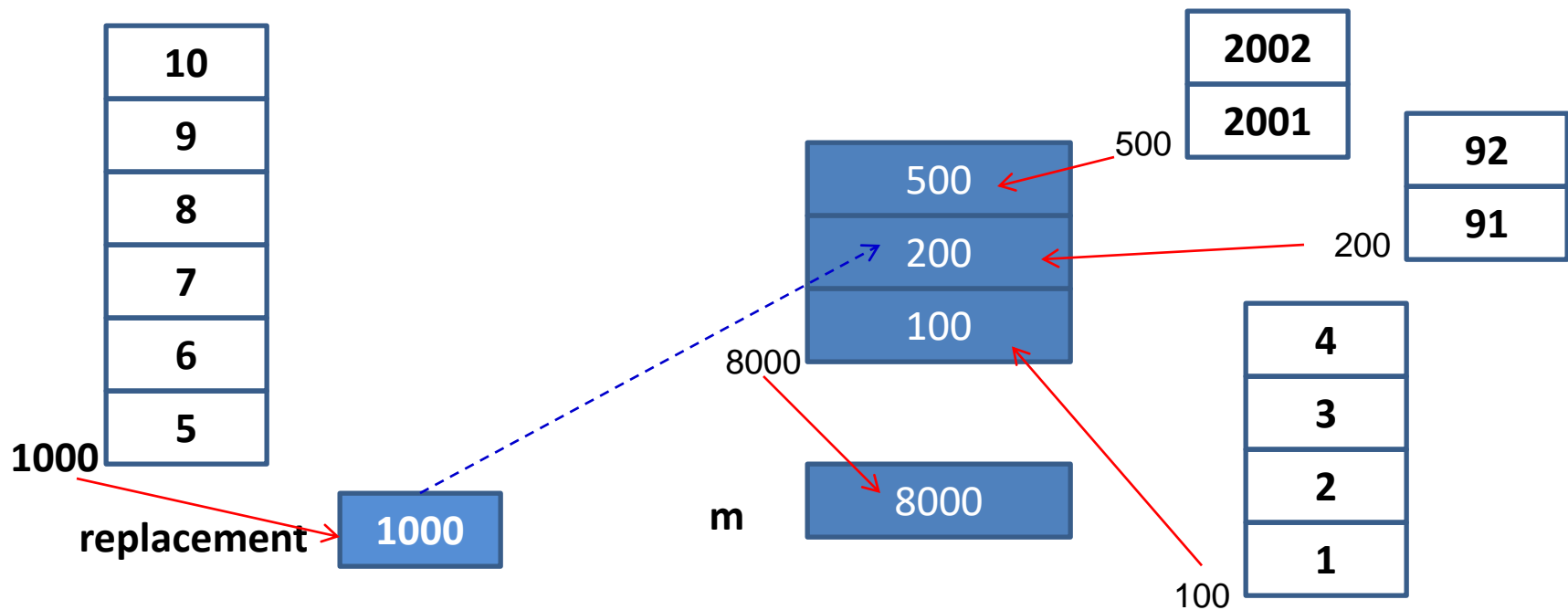- Copying Arrays
  - Use arraycopy method from System class.

One Dimensional Arrays (3)

int[] ar;

ar= new int[3];

ar[0]=1; ar[1]=2; ar[2]=3;

int a2[];

int[] a3 = {1,2,3,4,5};

int a4[] = {1,2,3,4,5};

Array is a reference variable

Heap

3

2

10000 | 1

Stack | ar | 10000

int n=10;

Multiple Dimensional Arrays

int m[][]= { {1,2,3,4}, {91,92}, {2001,2002}};
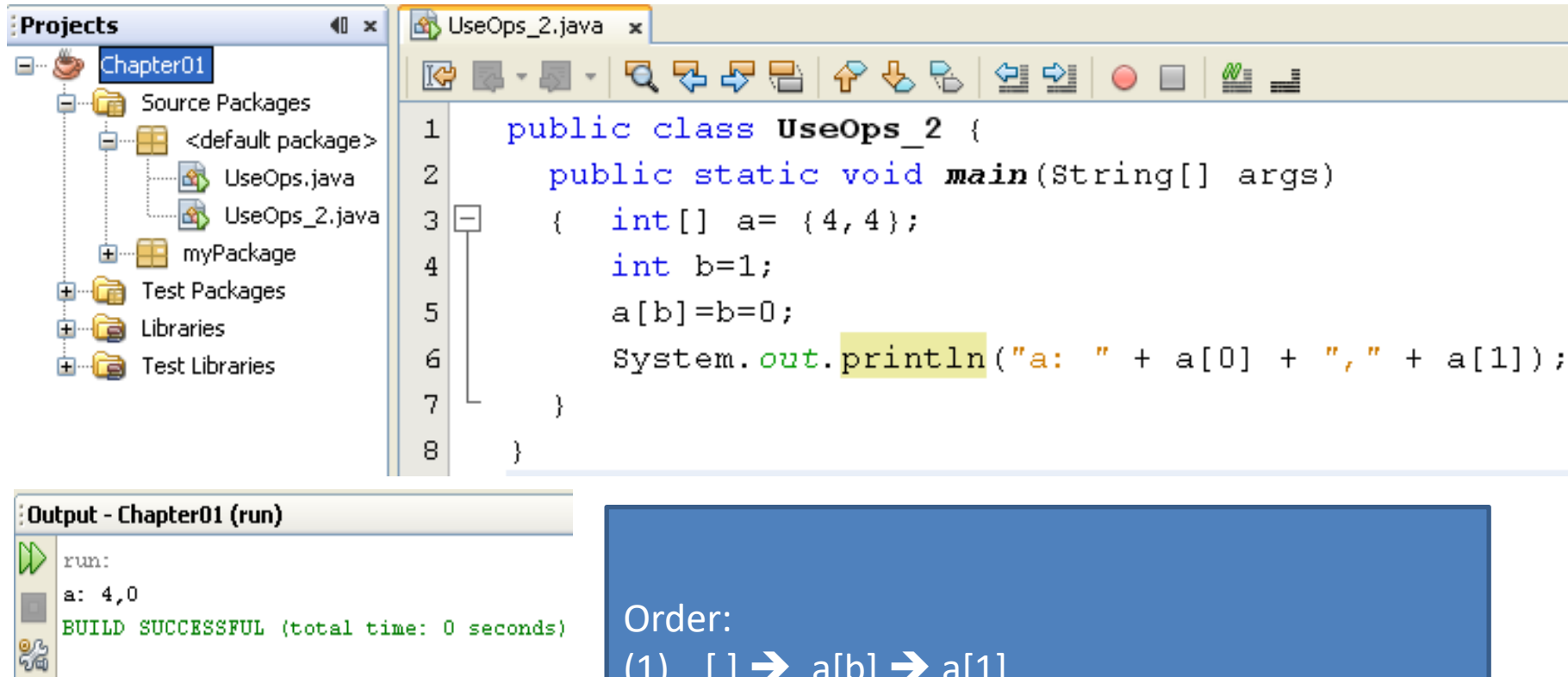int[] replacement = {5,6,7,8,9,10};
m[1]= replacement;

m[i][j]

```
int[][] m; // declare a matrix
int r=10, c=5; // number of rows, columns
m= new int[r][c]; // memory allocate
```

Evaluating Expressions and Operator Precedence

- The compiler generally evaluates such expressions from the innermost to outermost parentheses, left to right.

```
int x = 1; int y = 2; int z = 3;
int answer = ((8 * (y + z)) + y) * x;
would be evaluated piece by piece as follows:
((8 * (y + z) ) + y) * x
((8 * 5) + y) * x
(40 + y) * x
42 * x
42
```

Operator Precedence- Evaluation Order

```
1    public class UseOps_2 {
2        public static void main(String[] args)
3    {    int[] a= {4,4};
4        int b=1;
5        a[b]=b=0;
6        System.out.println("a: " + a[0] + "," + a[1]);
7    }
8    }
```

Projects
- Chapter01
  - Source Packages
    - <default package>
      - UseOps.java
      - UseOps_2.java
    - myPackage
  - Test Packages
  - Libraries
  - Test Libraries

UseOps_2.java

Output - Chapter01 (run)
```
run:
a: 4,0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Order:
(1) [ ] ➜ a[b] ➜ a[1]
(2) = (from the right) ➜ b=0 ➜ return 0
➜ a[1] = 0

Basic Constructs

- They are taken from C-language
- Selection

  if, if … else

  switch (char/int exp)… case … default…

- Loops

  for

  do… while

  while

Basic Logic Constructs

- They are the same with those in C-statements

```java
package com;
import java.lang.*;
public class Chao {
    public static void main(String args[]){
        System.out.println("Hello");
        int a[] = { 1,2,3,4,5};
        for (int i=0;i<a.length;i++)System.out.print(a[i] + ",");
        System.out.println();
        for (int x : a)  System.out.print(x + ",");
        System.out.println();
        for (int x : a) x+=10;
        for (int i=0;i<a.length;i++)System.out.print(a[i] + ",");
        System.out.println();
    }
}
```

An enhanced for loop

Read only

a | 1 | 2 | 3 | 4 | 5

x | 1

Output - P1 (run)

run:
Hello
1,2,3,4,5,
1,2,3,4,5,
1,2,3,4,5,

The String type

- A String represents a sequence of zero or more Unicode characters.
    - String name = "Steve";
    - String s = "";
    - String s = null;
- String concatenation.
    - String x = "foo" + "bar" + "!";
- Java is a case-sensitive language.

Type Conversions and Explicit Casting

```java
public class Casting_Convert_1 {
    public static void main (String[] args)
    {   short x, y = 256;
        byte m,n = 6;
        x = n ;  // Systematic Conversion
        n = y;   // narrow conversion
        n = (byte) y; // narrow casting, possible loss of precision
        System.out.println(n);
    }
}
```

```java
public class Casting_Convert_1 {
    public static void main (String[] args)
    {   short x, y = 256;
        byte m,n = 6;
        x = n ;  // Systematic Conversion
        n = (byte) y; // narrow casting, possible loss of p
        System.out.println(n);
    }
}
```

Output - Chapter04 (run)

```
run:
0
BUILD SUCCESSFUL (total time: 0 seconds)
```

* Widening Conversion: OK
• Narrowing conversion: Not allowed. We must use explicit casting.
• A boolean can not be converted to any other type.
• A non-boolean can be converted to another non-boolean type.

0000 0001

0000 0000

y

n

Scope of a Variable


```java
    public static void main (String[] args){
        int x=2, k=2;
        if(x<2){
            int y=3;
            int z=4;
        }
        y=6;
        for (int i=1; i<3; i++) x+=i;
        k+=i;
    }
}
```

Scope of the variable y

Scope of the variable i

Input/Output Data


```java
/* Write a program that will accept an array of intergers then
   print out entered value and the sum of values
*/
import java.util.Scanner;
public class InputOutputDemo {
    public static void main (String args[])
    {   int a[]; // array of integers
        int n ;   // number of elements of the array
        int i;    // variable for traversing the array
        Scanner sc= new Scanner(System.in); // object for the keyboard
        System.out.print("Enter number of elements: ");
        n = Integer.parseInt(sc.nextLine());
        a = new int[n]; // mem. allocating for elements of the array
        for (i=0;i<n;i++)
        {  System.out.print("Enter the " + (i+1) + "/" + n + " element: ");
           a[i]=Integer.parseInt(sc.nextLine());
        }
        System.out.print("Entered values: ");
        for (i=0;i<n;i++) System.out.format("%5d", a[i]);
        int S=0;
        for (int x: a)  S+=x;
        System.out.println("\nSum of values: " + S);
    }
}
```

- Class java.lang.System
- Class java.util.Scanner

Refer to Java documentation:
java.lang.String class,
- the **format** method,
  - format string
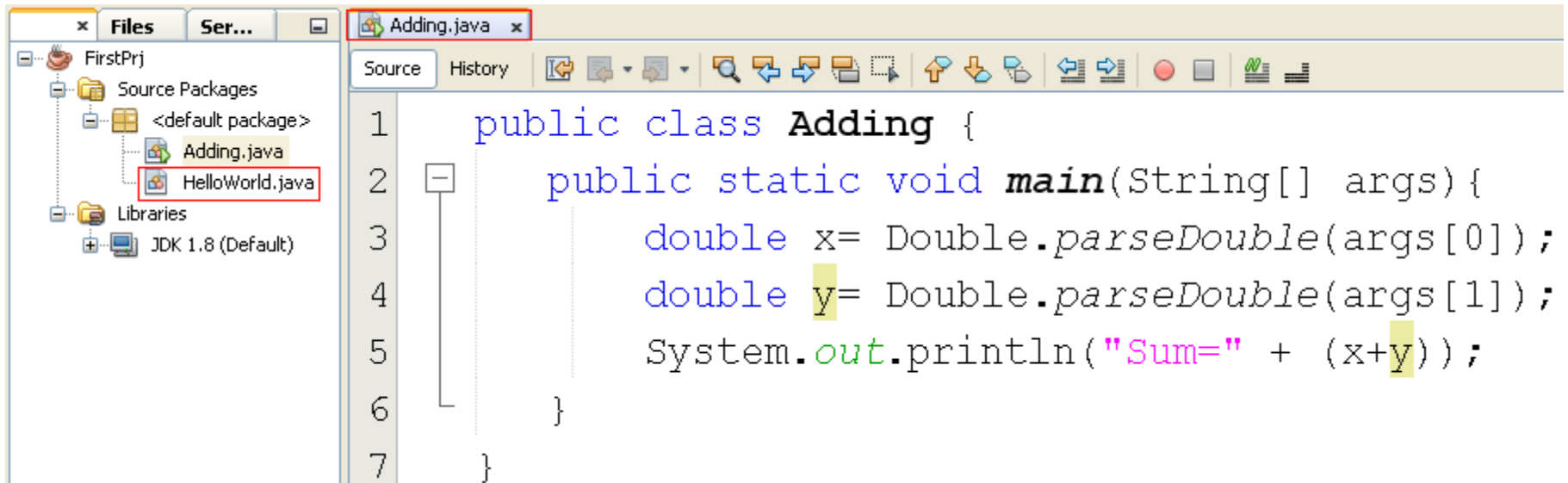for more details

n= sc.nextInt();

Output - Chapter01 (run) #2

```
run:
Enter number of elements: 5
Enter the 1/5 element: 1
Enter the 2/5 element: 4
Enter the 3/5 element: 2
Enter the 4/5 element: 0
Enter the 5/5 element: 7
Entered values:    1    4    2    0    7
Sum of values: 14
BUILD SUCCESSFUL (total time: 11 seconds)
```

Elements of Java Style

- Proper Use of Indentation
  - Statements within a block of code should be indented relative to the starting/ending line of the enclosing block.

- Use Comments Wisely

- Placement of Braces
  - Opening brace at the end of the line of code that starts a given block. Each closing brace goes on its own line, aligned with the first character of the line con.

- Descriptive Variable Names
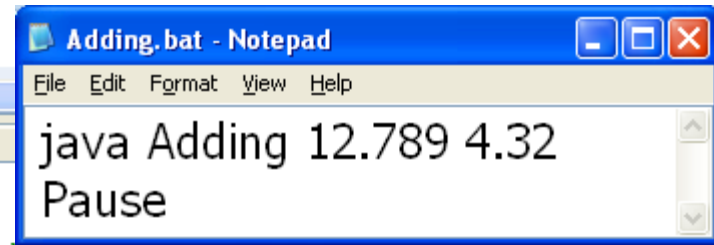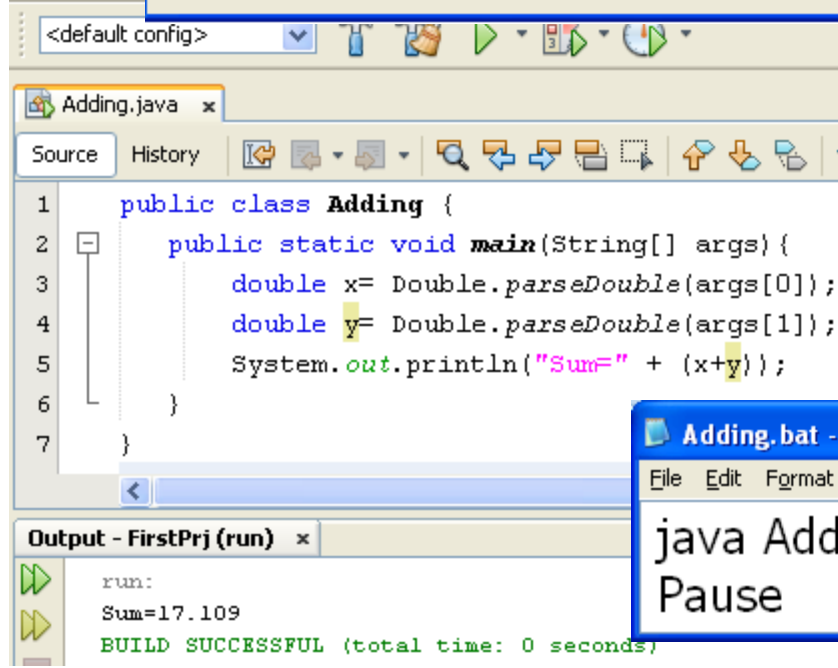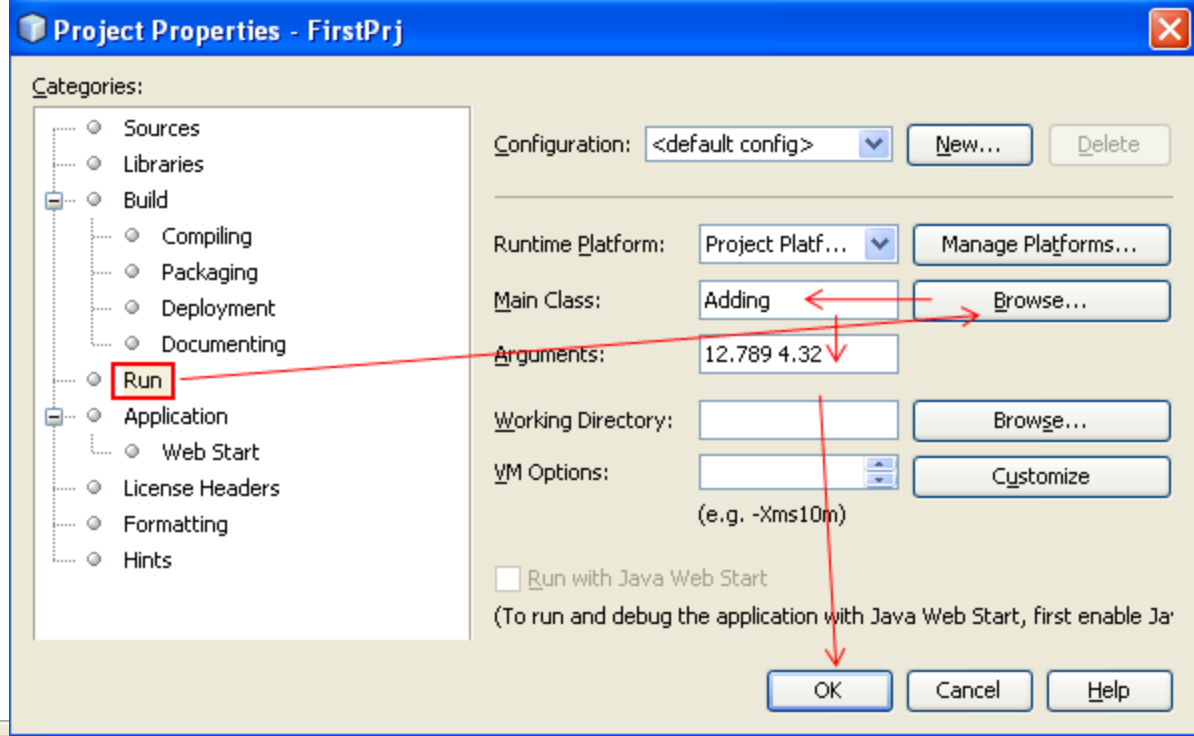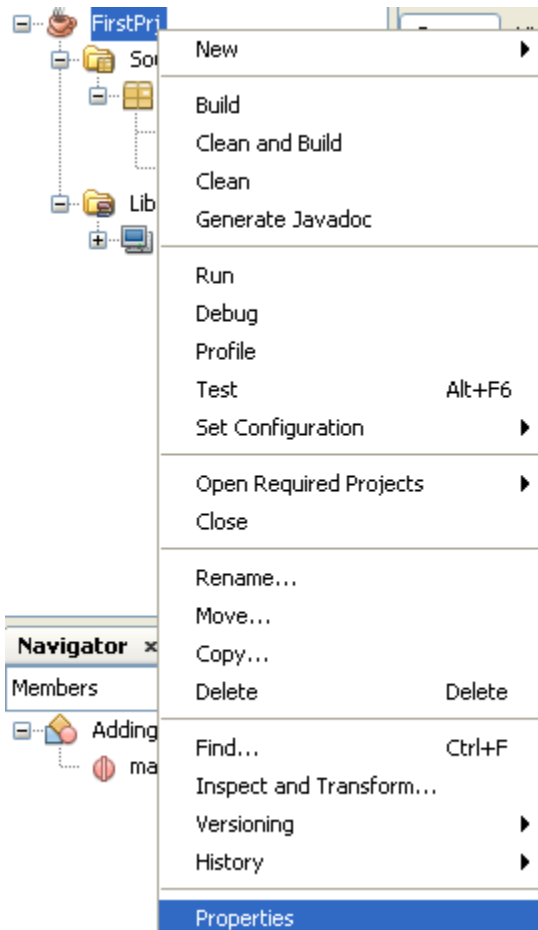
Pass Arguments to the method main


```java
public class Adding {
    public static void main(String[] args){
        double x= Double.parseDouble(args[0]);
        double y= Double.parseDouble(args[1]);
        System.out.println("Sum=" + (x+y));
    }
}
```

Pass Arguments to the method main

What Is a Package?

- A package is a namespace that organizes a set of related classes and interfaces.

- The Java platform provides an enormous class library (a set of packages) suitable for use in your own applications called API.

  - For example, a String object contains state and behavior for character strings.
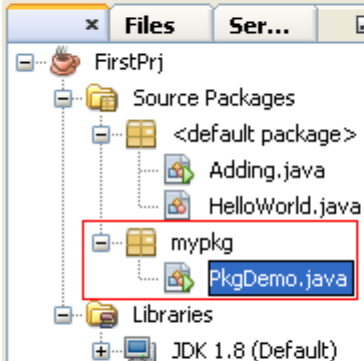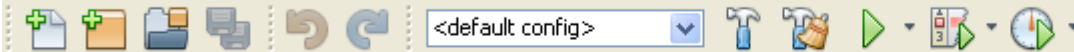
User-Defined Package

- **Add a Java class**

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: PkgDemo

Project: FirstPrj

Location: Source Packages

Package: mypkg

Created File: K:\GiangDay\FU\Java-OOP\Demos\FirstPrj\src\mypkg\PkgDemo.java

FirstPrj - NetBeans IDE 8.0.2

File   Edit   View   Navigate   Source   Refactor   Run

`<default config>`

Files   Ser...

- FirstPrj
  - Source Packages
    - `<default package>`
      - Adding.java
      - HelloWorld.java
    - mypkg
      - PkgDemo.java
  - Libraries
    - JDK 1.8 (Default)

Adding.java   PkgDemo.java

Source   History

```
1   package mypkg;
2   public class PkgDemo {
3       public static void main(String[] args){
4           System.out.println("Package Demo.");
5       }
6   }
7
```

If package is used, it must be the first line in Java code

Output - FirstPrj (run)

```
run:
Package Demo.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Summary

- The core concepts behind object-oriented programming: objects, interfaces, classes, and inheritance.

- The traditional features of the language, including variables, arrays, data types, operators, and control flow.